Curso teórico:
Orientação a Objetos

Matemática computacional
Marcos Aurelio Wozhiak Jr
webzhiak.com.br

Objetivos

- Conhecer os conceitos fundamentais de orientação a objetos;
- Aprender a criar e utilizar classes e objetos em C, C++, Java, PHP e outras linguagens de programação;
- Trabalhar de forma dinâmica com os conceitos aprendidos;
- Aplicar os conceitos de forma prática;
- Melhorar a eficiência de seus projetos;

Conceitos básicos

- Na programação orientada a objetos um programa de computador é conceituado como um conjunto de objetos que trabalham juntos para realizar uma tarefa.
- Cada objeto é uma parte do programa, interagindo com as outras partes de maneira específica e totalmente controlada.
- Na programação orientada a objetos, implementa-se um conjunto de classes que definem os objetos presentes no sistema de software.
- Cada classe determina o comportamento (definido nos **métodos**) e estados possíveis (**atributos**) de seus objetos, assim como o relacionamento com outros objetos

Classe

- A classe representa um conjunto de objetos com características afins.
- Uma classe define o comportamento dos objetos através de seus métodos, e quais estados ele é capaz de manter através de seus atributos. Exemplo de classe: Animais, Automóveis, Computador, etc...

- **Subclasse:** é uma nova classe que herda características de sua(s) classe(s) ancestral(is)

Objeto

- Um objeto ou uma instância de uma classe: é capaz de armazenar estados através de seus **atributos** e reagir a mensagens enviadas a ele, assim como se relacionar e enviar mensagens a outros objetos. Exemplo de objetos da classe Humanos: João, José, Maria;

Atributo

- Os atributo são características de um objeto.
- Basicamente a estrutura de dados que vai representar a classe.
- **Exemplos**:
  - *Funcionário*: nome, endereço, telefone, CPF,...;
  - *Carro*: nome, marca, ano, cor, …;
  - *Livro*: autor, editora, ano.
- Por sua vez, os atributos possuem valores. Por exemplo, o atributo cor pode conter o valor azul. O conjunto de valores dos atributos de um determinado objeto é chamado de estado.

Métodos

- Método definem as habilidades dos objetos.
- Bidu é uma instância da classe Cachorro, portanto tem habilidade para latir, implementada através do método de um latido.
- Um método em uma classe é apenas uma definição. A ação só ocorre quando o método é invocado através do objeto, no caso Bidu.
- Dentro do programa, a utilização de um método deve afetar apenas um objeto em particular;
- Todos os cachorros podem latir, mas você quer que apenas Bidu dê o latido.
- Normalmente, uma classe possui diversos métodos, que no caso da classe Cachorro poderiam ser sente, coma e morda;

Mensagem

- Mensagem é uma chamada a um objeto para invocar um de seus métodos, ativando um comportamento descrito por sua classe.
- Também pode ser direcionada diretamente a uma classe (através de uma invocação a um método estático);

Herança

- Herança é o mecanismo pelo qual uma classe pode estender outra classe ou, ainda, ser estendida de outra classe.
- O mecanismo de herança permite que uma classe (subclasse) compartilhe o código-fonte outra classe (superclasse), aproveitando seus comportamentos (métodos) e variáveis possíveis (atributos).
- A grande vantagem deste mecanismo, durante o desenvolvimento de uma aplicação, é evitar a duplicação desnecessária de código, o que pode levar a reduzir o tempo gasto para desenvolver o projeto.

Herança

- Generalização é o processo de herança, no qual é criada uma superclasse, a partir de subclasses já existentes.
- Especialização é o processo no qual é criada uma subclasse a partir de superclasse(s) já existentes. Neste último caso, a herança é simples, quando uma subclasse herda características de uma única superclasse;
- A herança é múltipla, quando a subclasse herda características de mais de uma superclasse.
- **Exemplo**: Mamífero é superclasse de Humano. Ou seja, um Humano é um mamífero.

Associação

- Associação é o mecanismo pelo qual um objeto utiliza os recursos de outro. Pode tratar-se de uma associação simples "usa um" ou de um acoplamento "parte de".
- Por exemplo: Um humano usa um telefone. A tecla "1" é parte de um telefone

Encapsulamento

- O encapsulamento consiste na separação de aspectos internos e externos de um objeto.
- Este mecanismo é utilizado amplamente para impedir o acesso direto ao estado de um objeto (seus atributos), disponibilizando externamente os métodos que acessam (getters) e alteram (setters) estes estados.
- **Exemplo**: você não precisa conhecer os detalhes dos circuitos de um telefone para utilizá-lo. A carcaça do telefone encapsula esses detalhes, provendo a você uma interface mais amigável (os botões, o monofone e os sinais de tom).

Abstração

- A abstração é a habilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais. Em modelagem orientada a objetos, uma classe é uma abstração de entidades existentes no domínio do sistema de software.

Polimorfismo

- Polimorfismo consiste no princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação (assinatura) mas comportamentos distintos, especializados para cada classe.
- De acordo com a literatura, existem quatro tipos de polimorfismo que uma linguagem de programação pode ter (atente para o fato de que nem toda linguagem orientada a objeto tem implementado todos os tipos de polimorfismo):

Polimorfismo

- **Universal:**
  - *Inclusão*: um ponteiro para classe mãe pode apontar para uma instância de uma classe filha (exemplo em Java: "List lista = new LinkedList();" (tipo de polimorfismo mais básico que existe)
  - *Paramétrico*: se restringe ao uso de templates (C++, por exemplo) e generics (Java/C♯)
- **Ad-Hoc**:
  - *Sobrecarga*: duas funções/métodos com o mesmo nome mas assinaturas diferentes
  - *Coerção*: a linguagem que faz as conversões implicitamente (como por exemplo atribuir um int a um float em C++, isto é aceito mesmo sendo tipos diferentes pois a conversão é feita implicitamente)

Interface

- A interface é um contrato entre a classe e o mundo externo. Quando uma classe implementa uma interface, ela está comprometida a fornecer o comportamento publicado pela interface

Pacotes / Namespaces

- Pacotes (ou Namespaces): são referências para organização lógica de classes e interfaces.